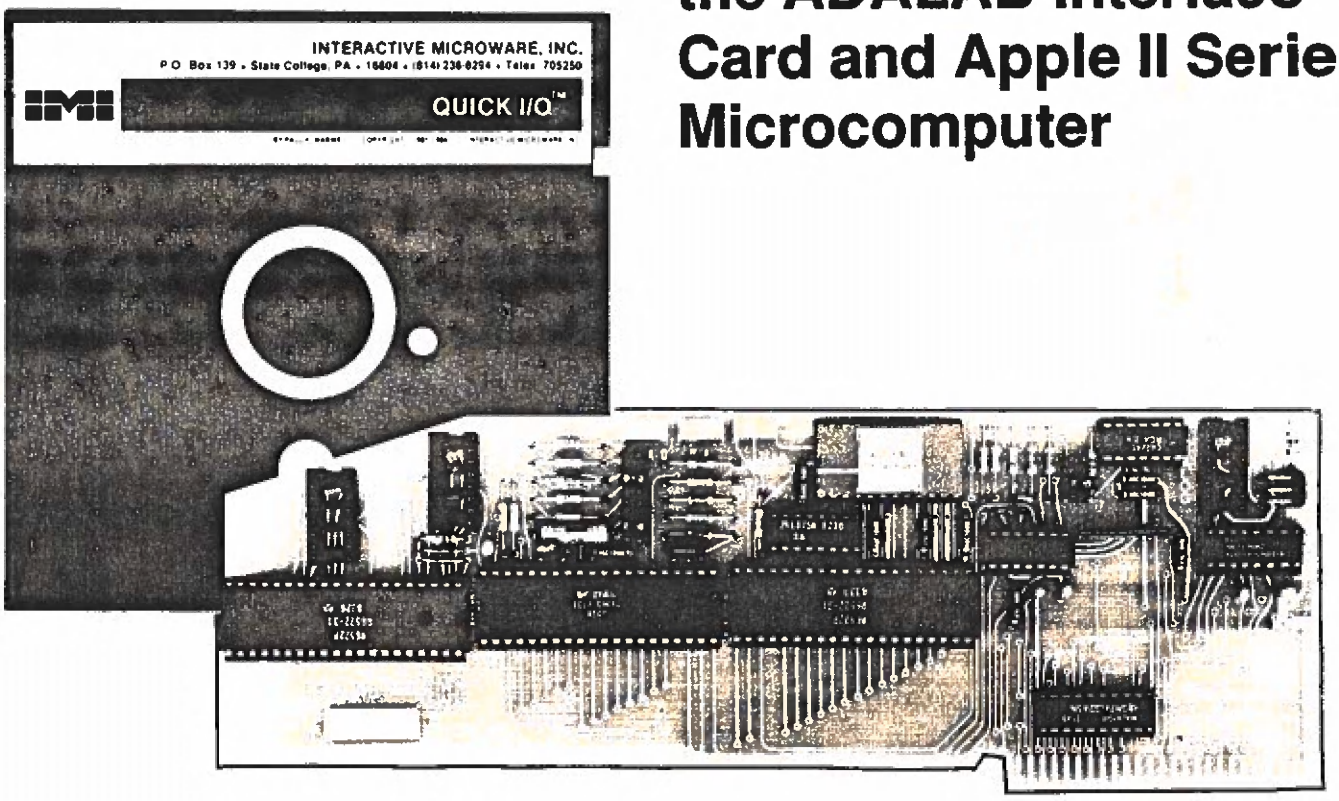
Instruction Manual

# General-PurposeData Acquisition Software For the ADALAB Interface Card and Apple II Series Microcomputer




QUICK I/0

*A Bioanalytical Systems Company*

TABLE OF CONTENTS

## COPYRIGHT NOTICE AND LICENSING AGREEMENT

This program is copyrighted, which means that it is illegal for anyone to make copies of it or give copies to others. However, the original purchaser of ADALAB and this program is hereby granted permission to make a backup disk for his own exclusive use on a single computer system. If you have purchased an ADALAB card, you may use QUICKI/O for any purpose. Resale or any commercial use of QUICKI/O requires written permission from Interactive Microware, Inc.

This disk also includes DIVERSI-DOS modifications to the operating system for faster disk operations. Therefore, we are required to include this copyright notice:

DOS 3.3 is a copyrighted program of APPLE COMPUTER, INC. licensed to INTERACTIVE MICROWARE, INC. to distribute for use only in combination with this product. APPLE COMPUTER, INC. makes no warranties, either express or implied, regarding the enclosed computer software package, its merchantability or its fitness for any particular purpose.

This disk also contains a high-speed operating system called Diversi-DOS (tm), which is licensed for use with this program only. To legally use Diversi-DOS with other programs, you may send $30 directly to: DRS, Inc., 5848 Crampton Ct., Rockford IL 61111. You will receive a Diversi-DOS utility disk with documentation.

## HOW TO PRODUCE A BACKUP DISK

The original QUICKI/O disk is recorded in DOS 3.3 format and may be copied by any standard disk copying program. Since your original copy is to be used only for archiving, the first thing you should do before using QUICKI/O is to make a backup disk as follows:

1. Insert the DOS 3.3 System Master disk provided by Apple Computer, Inc. and type RUN COPYA<RETURN>. Answer the questions about which disk drives are to be used for the original and duplicate disks.

2. Leave the write protect tab on the original QUICKI/O disk at all times, just to make sure that you don't accidentally erase it. Press the 'RETURN' key to begin copying. If you have two drives, the program will copy the disk unassisted. If you have a single disk drive, the program will prompt you when it is time to change disks. Insert the QUICKI/O disk whenever the program prompts:

   INSERT ORIGINAL DISK AND PRESS RETURN

Insert a blank disk whenever the program prompts:

   INSERT DUPLICATE DISK AND PRESS RETURN

3. If you wish to copy only the QUICKI/O object code file to another disk, insert the original QUICKI/O disk and type BLOAD QUICKI/O<RETURN>. Then, insert your copy disk and type BSAVE QUICKI/O, A$8D00, L$8F8<RETURN>.

## PREVIEW OF QUICKI/O FEATURES

QUICKI/O is one of IMI's software operating systems for the ADALAB data acquisition hardware card. Together with an Apple computer and Applesoft BASIC, they form a development system for data acquisition and control.

Using short, easy-to-use QUICKI/O commands from within Applesoft, any BASIC programmer with some experience in the lab can develop programs for a wide range of applications, from spectrophotometry to process control or even energy management for a building.

```
©1982 DSR C#254
371 FREE

*A 003 ADAAMP
*A 006 ADABYTEI/O
*A 002 ADASAMP
*B 002 MUXAD
*B 002 ADOBJ
*B 002 ADTIM
*A 003 BYTEST
*A 002 EX1
*A 002 EX2
*A 002 EX3
*A 002 EX4
*A 002 EX5
*A 002 EX6
*A 002 EX7
*A 004 EX8
*A 002 EX9
*A 002 EX10
*A 002 EX11
*A 003 EX12
*A 002 HELLO
*B 002 MUXOBJ
*A 025 QUICKSAMPLE
*B 010 QUICKI/O
*A 010 REMTROL TEST1
*T 002 TESTDATA
*B 002 TIMOBJ1
*B 002 TIMOBJ2
*A 004 TIMTEST1
*A 004 TIMTEST2
*A 003 ADAMUX TEST
*A 003 DATE
*A 009 QUIXTENDER
```

Figure 1. QUICKI/O disk file catalog

On the accompanying disk, you will find a demonstration program, labeled "QUICKSAMPLE", that is written using QUICKI/O and Applesoft. It is designed to give you a demonstration of Applesoft and QUICKI/O programming in all areas of the ADALAB's functions. It also provides test diagnostics for troubleshooting the ADALAB hardware. Also recorded on your QUICKI/O disk are twelve other short Applesoft and QUICKI/O examples which are referred to in this manual (see catalog of disk files in figure 1). There are also demonstration programs for various optional IMI hardware modules (REM-TROL, ADA-MUX, etc). Their use is described in the manual supplied with each module.

In order to use QUICKI/O, one or more ADALAB interface cards must be plugged into any slot of your APPLE computer. Up to four ADALAB cards may be used, and these will be called "card # 0" through "card # 3". You can then BLOAD QUICKI/O and it automatically locates each ADALAB card and "initializes" it. Thereafter, simple commands in BASIC enable you to control the ADALAB hardware, including the real-time clock and timers, Digital Input/Output (I/O), Parallel I/O and Analog I/O.

For example, &PI0 reads a value from the Parallel Input on ADALAB card # 0 and stores the value in the variable called D%. &PO0 sends the value in D% to the Parallel Output on card # 0. Another form of command allows you to transfer values to and from an array called D%(). For example, &PI0,5 reads a value from Parallel Input card # 0 and stores it in D%(5). &PO0,2 sends the value in D%(2) to the Parallel Output on card # 0. For ease of programming, the ADALAB card number and array element number may be replaced by any valid BASIC expression. For example, &PO(X/4), (Y+3) would transfer the value stored in D%(Y+3) to Parallel Output card # X/4. Commands such as these can be executed immediately after pressing RETURN, or they may be included in any BASIC program for repeated execution under program control. These commands are short so that they are easy to remember and convenient to type.

QUICKI/O is written in machine language so that commands are executed as rapidly as possible, typically in less than 0.002 seconds. In fact, there is no faster way to control your instruments, short of laboriously programming your entire application in machine language. QUICKI/O links your BASIC programs to the ADALAB hardware by a method that is very convenient and easy to learn. Also, since QUICKI/O uses only 2.25K of memory, you will have plenty of memory space left for your programs.

QUICKI/O can also provide a constant display of the time of day in hours, minutes and seconds in the upper right corner of the screen. You may set or read the current time from within your program, thus enabling you to start or stop external processes at particular times. In addition, a countdown timer can be set for any time interval. When the timer count reaches zero, the APPLE speaker will buzz to request your attention. It also continues to count, so that you can determine the amount of time that has elapsed since the alarm went off. The timer count may be read repeatedly at various intervals to an accuracy of 0.1 second. Thus, you can time more than one event under program control.

## HOW TO USE THE QUICKSAMPLE DEMONSTRATION PROGRAM

Before using QUICKSAMPLE, you should install the ADALAB card as described in the ADALAB hardware manual. The following explanation assumes that you have connected all three cables to the analog junction box and that the A/D and D/A jumpers are both at ±1 volt. It is also possible to use some parts of the QUICKSAMPLE program with other inputs and outputs from instruments, etc. Feel free to list the QUICKSAMPLE program and use any portion of it to develop applications for your own personal use.

NOTE: If you install ADALAB in slot 3 and you are using an Apple 80-column card, you must first POKE 49163,0 to enable the ADALAB card. To re-enable the 80-column card, POKE 49162,0.

Insert the QUICKI/O disk and turn on your Apple Computer or type RUN QUICKSAMPLE if the computer is already turned on. This program automatically loads QUICKI/O and initializes the ADALAB hardware. The screen will now display the main menu, as follows:

```
****************************************
        QUICKI/O SAMPLER OPTIONS:

****************************************
  1=SET TIME          2=READ TIME

  3=SET ALARM         4=READ ALARM

  5=DIGITAL I/O       6=GENERAL I/O

  7=LINEARITY TEST    8=STABILITY TEST

  9=SELF TEST         0=STOP

****************************************
          ENTER OPTION (0:9)
```

Enter 9 <RETURN> after ENTER OPTION (0:9)? for the initial self test on ADALAB card # 0.

The self test program:

1. starts the real time clock in the top right hand corner of the screen,
2. verifies that alarm (timer 1) is working,

3. runs a test pattern through the Digital I/O and Parallel I/O, and
4. sends a series of voltages from the D/A converter to the A/D converter.

The voltages output from the Digital to Analog (D/A) converter are read by the Analog to Digital (A/D) converter and a running tally of the difference between the D/A output value and the A/D input value is printed on the screen.

> NOTE: Although each ADALAB card is calibrated and tested carefully at the factory, your first use of QUICKSAMPLE and the ADALAB may not give optimal results. This is because each ADALAB must be calibrated in the Apple it will be used in, due to some variability in power supply voltages. See the ADALAB hardware manual for instructions on calibrating ADALAB.

On the bottom line of the screen, the present D/A output value is printed in the column labelled VOUT=, the present A/D input value is labelled VIN=, the difference is labelled ERROR= and the maximum difference between VOUT and VIN is labeled MAX ERROR=. A separate number is given for the maximum positive and the maximum negative difference between VOUT and VIN. For example, if the largest positive error is +4 and the largest negative error is -3, then MAX ERROR= -3/+4.

If you have plugged the ADALAB cables properly into the analog junction box, all of these tests should proceed without any problems. The real-time clock test, the timer 1 test, the digital I/O test and the parallel I/O test should each print OK as the last thing on the corresponding line. The analog I/O test should display a maximum error of +10/-10. If ERROR is printed instead of OK after any test or if the analog error is greater than +5/-5, you should consult the ADALAB hardware manual section 4.0, CALIBRATION PROCEDURES before proceeding. The hardware manual describes a procedure for calibrating the A/D and D/A converters.

After these tests have been completed, you may change the voltage being output by the D/A converter by repeatedly pressing the left or right arrow keys. The left arrow key decreases the voltage, and the right arrow key increases the voltage. The first time you press one of the arrow keys, the step size is one; however, the step size increases by one each time you press the same arrow key. This will allow you to quickly scan to any possible output value and read the corresponding input value. On the Apple II+, you may press the REPT key, together with one of the arrow keys, in order to speed up the scanning. On the Apple IIe, just hold down the arrow key to repeat. Whenever you switch from one arrow key to the other, the step size goes back down to one; thus, you can narrow down on a particular output voltage.

To exit from the self-test of ADALAB card # 0, press any key

other than one of the arrow keys.

1 = SET TIME

After the self-test has finished, the screen will be erased and the main menu will re-appear on the screen. To start the real time clock, select the first option by typing a 1. Then, enter the current time as a 3- or 4-digit number, using a 24 hour format. For example, if the time is 4:30 p.m., type 1630. You will notice that the time is now displayed in the upper right corner of the screen and the time is updated once each second.

2 = READ TIME

Next, you might want a report of the time. Of course in this case you could just look at your watch or look at the top of the screen, but in order to demonstrate the code in QUICKSAMPLE which can perform this function for you, select option 2.

After this program option is completed, you will be asked to press RETURN. This gives you time to read the results before they are erased. After you press RETURN, the screen will be erased and the menu will be redisplayed.

3 = SET ALARM

Option 3 is used to set the countdown timer/alarm. If you have more than one ADALAB card, you must enter an ADALAB card number at this point; type 0 for the first card, 1 for the second, and so on. Then, you will be asked to enter a time interval in units of 0.1 second; for example, if you want the alarm to buzz after 3 seconds, type 30.

4 = READ ALARM

After you set the timer alarm, you can read the time remaining by selecting option 4. If you repeatedly select option 4, you will find that it counts down at the rate of 10 per second, regardless of whatever else the program may be doing.

Reading the timer/alarm disables the buzzer; thus, if you wish to hear the alarm, don't read the timer until it starts buzzing. You will note that the timer/alarm count is negative after the buzzer begins; this tells how long it has been since the alarm went off (in units of 0.1 seconds).

5 = DIGITAL I/O

Now, let's test the digital input/output by selecting option 5. The screen display shows the state of each input bit or line

(blank for off and * for on) and it also shows the state of each output bit (0 for off and 1 for on). The bits are numbered from 7 (the most significant bit) to 0 (the least significant bit). On the Apple keyboard, keys 0 through 7 will reverse the state of the corresponding output bit and its displayed status. For example, if bit 2 is off (0), you can turn it on (1) by typing 2. When the ADALAB cables are connected to the analog junction box (AJB), the digital outputs are directed right back into the digital inputs. Therefore, when using the AJB, the displayed digital input states should always be the same as the digital output states.

6 = GENERAL I/O

Option 6 selects the general purpose I/O test. First, you will be asked to select a source of input data.

```
INPUT DEVICES ARE:
1=KEYBOARD  2=DATA BUFFER
3=PARALLEL  4=ANALOG
INPUT OPTION (1:4)?
```

To type in data from the keyboard, type 1<RETURN>. The input data are always stored in a data buffer (array D%); thus, after your first use of option 6, you may reuse the same input data by selecting input option 2. Options 3 and 4 select parallel or analog to digital converter input.

Next, you will select the output device. Your screen should look like this:

```
OUTPUT DEVICES ARE:
1=SCREEN LIST  2=GRAPH
3=PARALLEL  4=ANALOG
OUTPUT OPTION (1:4)?
```

Option 1 causes the input values to be printed on the screen.

Option 2 plots the input values using high-resolution graphics; in this case, you will be asked to enter a maximum and minimum value. The program will then calculate a scale factor that each value will be multiplied by before it is plotted.

Option 3 will transmit data to ADALAB's parallel output port.

Option 4 will allow you to vary the analog voltage output by the D/A converter.

Next, you will be asked to set some sampling parameters. Assuming that you have selected keyboard input (option 1), the questions should look like this:

```
# OF VALUES (1:5000)?
DELAY TIME (IN TENTHS OF SECONDS)?
VALUE 1?
PRESS RETURN TO CONTINUE
```

The first question asks how many values (1 to 5000) you wish to have input and output. Next, you are asked to select a delay time between inputs, in units of 0.1 seconds. For example, if you want to input a value once each second, type 10. Normally, you should enter a delay time of 0 (that is, no delay between samples) for maximum speed, but do not use a delay time of 0 for analog input, as this will not work properly at this speed.

As an example, let's input some values from the keyboard and transmit them to the Analog Output. After starting the program, select option 6 (GENERAL I/O) on the main menu. Then, enter 1 to select the keyboard for input. Next, enter 4 to select Analog Output. If you have more than one ADALAB card, you will be asked to enter an ADALAB card number. When it asks HOW MANY VALUES?, type 10. Since our typing speed is much slower than the Analog Output can handle, you may type 0 when it asks for the DELAY TIME.

The values you type in next should represent the number of counts for the voltage you wish to send out through the D/A converter. Now the program will ask you to type in your VALUE 1, VALUE 2 and so on. After you type each value, it will be sent immediately to the Analog Output on the card selected. You may connect a voltmeter to the voltage test points on the analog junction box in order to verify that the Analog Output is working properly. If you type the value -2047, the Analog Output will give the most negative voltage for the range you have selected (see the ADALAB hardware manual for more information about jumper options). If you type 0, the output will read zero volts. If you type 2047, the voltage will be the most positive value for the selected range. For example, if you type in 1023, and your D/A converter is set on its 1 volt range, the D/A converter should output 0.4999 volts.

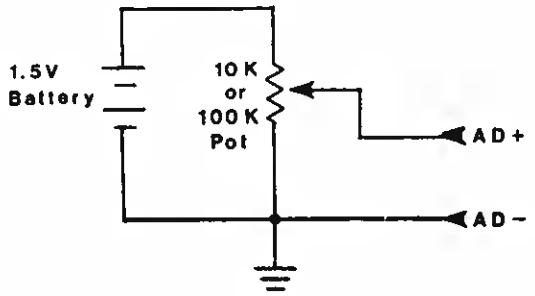
Our second example will collect samples from the Analog Input on card # 0 and plot a graph. After starting the program, select option 6 (GENERAL I/O) on the main menu. Then type 4 to select Analog Input. If you have more than one ADALAB card, you must enter a card number. Next, select option 2 for GRAPHic Output. The program will plot the maximum negative voltage at the bottom of the screen, 0 volts in the center and the maximum positive voltage at the top of the screen. Then, enter 50 or more for the number of VALUES and enter 1 to select a DELAY TIME interval of at least 0.1 seconds between samples. After you type the delay value and press RETURN, the screen will be erased and a series of points will be plotted. If the analog cable is plugged into the analog junction box, the input voltage will be constant (whatever you last sent out on the Analog Output), so the plot should be a straight line. In fact, this is a good way to check the stability of the analog circuits on the ADALAB board. Since

the plot is scaled down quite a bit, you might want to list the values stored in the data buffer in order to detect small changes in the Analog Input values that are not visible on the graph. To do this, select option 2 (DATA BUFFER) for input and select option 1 (SCREEN LIST) for output.

If you wish, you may connect some other voltage source to the Analog Input on card # 0. The plot will then be a curved line that follows the applied voltage.

IMPORTANT: Before connecting anything to the banana jacks on the analog junction box, you must remove the black plastic jumpers connecting AD+ to DA and connecting AD- to GND (analog ground).

A good source of simulated instrument signals for testing the Analog Input is a battery connected to a potentiometer as shown below:




7 = LINEARITY TEST

Option 7 on the main menu tests the Analog Input and Analog Output circuits to make sure that their signal response is linear over their entire voltage range. Before running this test, plug the analog I/O cable into the analog junction box so that the Analog Output is connected to the Analog Input. The two plastic jumpers provided should connect pins DA to AD+ and AD- to GND on the AJB. Also, make sure that the voltage range jumpers on the A/D and D/A converters on ADALAB are on the same range. If you have more than one ADALAB card, the program will ask you to enter a card #.

During the analog linearity test, a graph of the input voltage will be plotted; it should be a straight line. After the last point is plotted, a straight line will be drawn from the last point to the first point for comparison with the plotted points. Then, the maximum differential nonlinearity will be determined by computing the maximum difference between successive pairs of input values. Since the output value changes in steps of 15, the difference between successive input values should also be 15. However, a maximum differential nonlinearity of ±2 (0.05% of full scale range) is considered to be within specifications for the ±4V range. Somewhat greater nonlinearity may be observed at higher gain values.

The maximum integral nonlinearity will also be reported. This is calculated by comparing each input value with the predicted value, based on the average step size over the entire voltage range. A maximum integral nonlinearity of +4/-4 (±0.1% of full scale range) is considered to be within specifications for the ±4V range. Note that these nonlinearity figures include errors from four possible sources: the D/A converter, the output amplifier, the input amplifiers for the A/D converter and the A/D converter itself. By testing the analog I/O in this way, we tend to exaggerate the nonlinearity error.

## 8 = STABILITY TEST

Option 8 on the main menu permits you to measure the short term and long term stability of the analog I/O subsystems. Before running this test, plug the analog I/O cable into the AJB and install jumpers from DA to AD+ and from AD- to GND on the AJB. Normally, the voltage range jumpers on ADALAB should be on the same range for the D/A as for the A/D, although this test will operate successfully with different ranges, provided that the D/A output voltage is within the range of the A/D input.

At the beginning of this test, you will be asked to enter the card number if you have more than one ADALAB card. You will now see:

```
OUTPUT VOLTAGE (-2047 : 2047)?
```

Enter the output voltage, which should be a value between -2047 and 2047. After this, the screen will be erased and a scale from -5 to +5 will be printed along the bottom.

The three lines of information below the stability graph will look something like this:

```
CARD#=0  VOUT=-60  V0=-60  T0=229
SCALE=1  ERROR%=0  V1=-59  T1=229
PRESS RETURN TO STOP STABILITY TEST
```

The card number, D/A output value (VOUT), initial A/D value (V0) and the initial time (T0) are printed on the first line. On the second line are printed the SCALE factor being used for the histogram graph; the percentage of input values which fall outside the ideal range of ±1 (ERROR%); the current input value (V1) and the current time (T1). A continuous stream of input values is read by the A/D converter, and a histogram of the deviations from the average input value (V1) is plotted. For example, if an input value of 1002 is read from the A/D while the average value (V1) is 1000, the +2 column on the histogram will be incremented. If one of the columns runs off the top of the screen, the SCALE factor will be incremented and the histogram will be replotted according to the new SCALE factor.

Short term stability of the analog I/O subsystem is

indicated by the distribution of values on the histogram and by the ERROR% figure on line 3. Bear in mind that V1 is the average of the most recent batch of 15 A/D input values and therefore, the histogram reflects only the short term stability. Long term stability (drift) is indicated by the difference between V0 and V1, over the time interval T0 to T1. You will note that the ERROR% value generally starts out high, because it takes a few readings before the A/D input settles to the exact value. Thereafter, the ERROR% value decreases rapidly. Also, note that ERROR% is not a percentage of full scale voltage; it is actually the percentage of values falling outside the ideal of ±1 least significant bit. This is a very sensitive standard of accuracy.

Long term drift is primarily a function of temperature. If you have just turned on the computer, you may expect a long term drift of as much as 10 counts over the first half hour of warmup, but thereafter, the drift will be much less. Removing the cover of the computer or adding cooling fans after calibrating will also cause drift because the drafts of air will change the temperature of the interface card. In many applications, long term drift is of little significance because measurements are made over a short time interval or else the application program can measure and compensate for drift.

## 9 = SELF TEST

Option 9 on the main menu selects the self-test program. As discussed earlier, the self-test program checks all parts of the standard ADALAB hardware.

## BACKGROUND INFORMATION ABOUT QUICKI/O PROGRAMMING

### Initialization of QUICKI/O

QUICKI/O is a machine language program that quickly executes commands that you may include in your BASIC programs. The easiest way to activate QUICKI/O for use in immediate (keyboard command) mode is to type BRUN QUICKI/O. This initializes the ADALAB hardware and then returns you to BASIC. Thereafter, you may type QUICKI/O commands (see below) in the immediate mode and obtain your results immediately. If you have already used BASIC to run another program, type NEW to delete all variables and arrays before you attempt to use any immediate mode commands. (D% must be the first variable defined after NEW.)

To use QUICKI/O in deferred execution (programming) mode, you must begin every BASIC program that wishes to use QUICKI/O commands with the following statements:

```
1   HIMEM: 36095: D%=0: DIM C%(5), Q%(5), D% (desired size):
      PRINT CHR$(4)"BRUN QUICKI/O"
```

Notice that D% must be the first variable named in your program. When this statement is executed, QUICKI/O is loaded at address $8D00 through $95FF and the ADALAB hardware is initialized. QUICKI/O automatically determines which slots contain an ADALAB card. The card in the lowest-numbered slot will be called card # 0, the card in the next higher-numbered slot is card # 1, and so on. Memory location 36221 ($8D7D) contains the number of ADALAB cards found. Memory locations 36222 to 36225 tell which slots are occupied by ADALAB cards (192 is added to each slot number in this list and the last value is set to zero.)

If you want to avoid having to reload QUICKI/O each time you re-run your program, you may change line 1 as follows:

```
1  HIMEM: 36095: D%=0: DIM C%(5), Q%(5), D% (desired size)

2  CALL  36096: REM INITIALIZE HARDWARE
```

In this case, QUICKI/O will only be loaded and initialized the first time you run your program.

### QUICKI/O Command Format

QUICKI/O commands are very short, easy to type and easy to remember. The first letter of every command is the ampersand (&); this tells BASIC that a QUICKI/O command is to follow. The second letter of each command selects the type of device; T selects the Real-Time clock or a Timer, D means Digital, P means

Parallel and A means Analog. The third letter of a QUICKI/O command is either I for Input or O for Output. The fourth letter selects the ADALAB card number; this may be a number, a variable name or an expression that represents the card number. Let's consider a few examples: &PI0 means Parallel Input on card # 0. &DO1 means Digital Output on card # 1. What does &AI0 mean? What does &TO2 mean? Now, suppose that the variable X has a value of 2. What does &DOX mean? What card would be read by the command &PIX/2? [ANSWERS: &AI0 means Analog Input on card # 0, &TO2 means Timer Output on card # 2, &DOX means Digital Output on card # 2 and &PIX/2 means Parallel Input on card # 1].

By now, you're probably wondering where the values for input and output are placed. Variable D%, the first variable defined in line 1, is used for both input and output. In other words, the result from an Input command is returned in D% and you should place the desired output value in D% before issuing an Output command. Another form of command allows you to input and output values from the array named D%(). To do so, your QUICKI/O command would be followed by a comma and a number, variable name or expression that represents the element number in D%() that contains the input or output value. For example, the command &AI0,10 returns a value in D%(10) and &PO0,10 outputs the value in D%(10). If the value of X is 10, &AI0,X and &PO0,X would give the same results as &AI0,10 and &PO0,10.

## QUICKI/O Error Conditions

If an error is detected in any QUICKI/O command, a SYNTAX ERROR will result. Any of these conditions will cause an error:

1. If line # 1 (above) is missing.
2. The second command letter is not T,D,P or A.
3. The third command letter is not I or O.
4. The ADALAB card number is larger than the number of ADALAB cards in your computer.
5. The element number is larger than the DIMension of D%().
6. An invalid expression is given for the card number or element number.
7. QUICKI/O has not been loaded or initialized.

If such a SYNTAX ERROR occurs in a running program, BASIC will print the line number in which the error occurred. In immediate mode, no line number will be printed.

## Other Things You Should Know About QUICKI/O

After QUICKI/O has been initialized by executing line # 1 (above), commands may be executed in deferred mode (while your program is running) or in immediate mode (when your program is stopped). Bear in mind that input values are returned in variable D% or array D%(), and also that any values to be output must be placed in variable D% or array D%(). IMPORTANT: D% must

be the first variable declared in your program. If you want to use array D%, you must DIMension C%(5), Q%(5), D%(desired size) as the first arrays declared in your program.

Arrays C% and Q% are not actually used currently by QUICKI/O, but they must be included to allow for upward compatibility with present and future IMI software. If you do not declare D% as your first variable or declare the arrays as described, QUICKI/O will not be able to find these variables!

After you start the clock, it continues to tick along, even if you stop your program. You can even load and run a different program without stopping the clock, but in this case you must make sure that the new program doesn't erase any part of QUICKI/O (set HIMEM:36095). Since QUICKI/O uses memory locations 36096 to 38399, you should press RESET before running any program that uses memory above 36095.

If you press RESET, the clock stops and the ADALAB hardware is partially disabled. To reinitialize the hardware, type CALL 36096. The CTRL-RESET Vector (at $3F2, $3I3) has been changed by QUICKI/O and now automatically takes care of disabling interrupts.

## DETAILS AND EXAMPLES OF QUICKI/O FUNCTIONS

One very good way to learn how to use QUICKI/O is to study the code used in the QUICKSAMPLE demonstration program. To obtain a listing of this program, just LOAD QUICKSAMPLE, activate your printer (using the PR#X command, where X is the printer card slot) and then type LIST. The code for options 1 through 9, respectively, will be found at the subroutine addresses following the ON . . . GOTO command in line 1150. The example programs discussed in the remainder of this manual will also help you to learn how to use all features of QUICKI/O.

### Setting the Real-Time Clock

Timer 0 is used as the real-time clock. It is very accurate because it is controlled by the quartz crystal oscillator in your APPLE. To set the time, set D% equal to the hour times 100 plus the minutes, using 24 hour format, with no colon. For example, if the time is 4:30 p.m., set D% to 1630. Then, type &TOO. When you set the time, the seconds count is always reset to zero. Thus, if you want the time to be accurate to the nearest second, wait until the seconds reading on your watch reaches zero before typing <RETURN> following the &TOO command. After you set the real-time clock, the time will be displayed in hours, minutes and seconds in the top right corner of your display screen. The display is updated each second. If interrupts are disabled by your program, the clock will fall behind; however, the clock will catch up as soon as interrupts are reenabled. Since the disk operating system (DOS) disables interrupts while reading or writing, you may notice the clock falling behind temporarily during disk operations.

### How to Disable or Enable Interrupts

When the real-time clock is running and/or whenever interrupts are enabled for any other device, conflicts with the Disk Operating System (DOS) may occur at random times. This is because DOS sometimes uses page zero location $45, which is also used to store the A register contents when an interrupt occurs. The only way to prevent this conflict is to disable interrupts before executing any DOS command, including the "PR#" command. After the DOS command is finished, interrupts may be reenabled. The real-time clock will recalculate the correct time if interrupts are disabled for less than 6553 seconds (1.82 hours). However, no data will be input or output from other devices that depend on interrupt servicing as long as interrupts are disabled.

The following short machine language subroutines have been included in QUICKI/O to disable or enable interrupts:

| LOCATION | INSTRUCTION | |
|---|---|---|
| $95F0 (38384 decimal) DISABLE | SEI | ;$78=120 decimal |
| | RTS | ;$60=96 decimal |
| $95F2 (38386 decimal) ENABLE | CLI | ;$58=88 decimal |
| | RTS | ;$60=96 decimal |

From BASIC, you should CALL 38384 to disable interrupts before using a DOS command. After the DOS command is finished, you should CALL 38386 to re-enable interrupts if you want to restart the real time clock. For example, to issue the PR#0 command to DOS, you should include these statements:

```
CALL 38384: PRINT CHR$(4)"PR#0": CALL 38386
```

## Reading the Real-Time Clock

The command to read the real-time clock is &TI0. This returns the time in D%, in the same format as you used to set the clock. In other words, if the time is 10:30, D% will be set to 1030. After the hour reaches 12, it is not reset to zero, but continues counting up to 13, etc. as they do in the military.

NOTE: The current time in hours, minutes and seconds is stored by QUICKI/O at locations 36201 ($8D69), 36190 ($8D5E) and 36179 ($8D53), respectively. You may PEEK these locations to read the time. Also, location 36178 (&8D52) contains a value that counts from 0 to 19 at the rate of 1 count every 50 milliseconds.

## Examples Using the Clock

The QUICKI/O disk includes many short example programs for you to test and/or modify for your own use. The examples are numbered in consecutive order to follow the examples in this manual.

The first example on the disk is EX1. Type RUN EX1 to see this example illustrate the setting and reading of the clock. You may press <CTRL>C to stop this program prematurely.

```
0   REM EXAMPLE 1
1   HIMEM: 36095: D% = 0: DIM C%(5), Q%(5), D% (desired
       size): PRINT CHR$(4)"BRUN QUICKI/O"
10  INPUT "WHAT TIME IS IT?";D%: & TO0
20  INPUT "WHEN DO YOU WANT TO STOP?";T
30  & T10: IF D% < T GOTO 30
40  PRINT "IT IS "D%"!  TIME TO STOP"
50  STOP
```

ANALYSIS:

Line 10 sets timer 0 to the time that you enter. For example, if the time is 4:30, enter 430 (without a colon).

Line 20 asks you to enter the enter stop time, in the same format.

Line 30 continues to read the time if the current time is less than the stop time.

Line 40 prints the time when the current time equals the stop time.

## The Countdown Timer/Alarm

Timer 1 is a 16-bit timer/alarm. To set the timer, place a value in D% and then type &TO1. This will set the timer to the value you have previously placed in D%. To read the alarm, issue the command &TI1. This returns the timer value in D%. The timer continuously counts down at the rate of 10 counts per second, regardless of whether you set the timer or not. However, if you have set the alarm, the speaker will start to buzz when the count reaches zero and will continue to buzz as long as the count is negative. The buzzing will stop when you read the alarm. Note that if you read the timer while the count is still positive, the buzzing will not occur, although the timer will continue to count down.

The countdown timer is very useful for timing various activities and for starting or stopping something after a certain delay.

## Examples Using the Countdown Timer/Alarm

The following program measures the length of time that Digital Input bit 1 is off. Note that the digital bits are on (logic 1) when nothing is connected. To turn bit 1 off, connect pin 8 (ground) to pin 16 on the digital input cable. Be very careful not to short out any other pin in doing so! Type RUN EX2 for this program.

```
0  REM EXAMPLE 2
1  HIMEM: 36095: D% = 0: DIM C%(5), Q%(5), D% (desired
     size): PRINT CHR$(4)"BRUN QUICKI/O"
10 D% = 32767: & TO1
20  & DI1: IF D% > 0 GOTO 20
30  & TI1: T = D%
40  & DI1: IF D% = 0 GOTO 40
50  & TI1: PRINT "DURATION = "(T - D%) / 10" SECONDS"
```

ANALYSIS:

Line 10 sets timer to the maximum value.

Line 20 waits until bit 1 goes off (D%=0) before starting.

Line 30 reads the start time, T.

Line 40 cycles through line 40 until bit 1 is on (D% > 0).

Line 50 reads the time when the bit went on and prints the duration.

This program will work for time intervals as long as 6553.5 seconds (1.82 hours), but beyond this time, the duration will start over at 0.

In the next example (EX3 on disk), we wish to collect 10 samples from the Analog Input on card # 0 with a certain time interval (DLAY) between each sample. The sample values will be stored in array D%().

```
0   REM EXAMPLE 3
1   HIMEM: 36095: D% = 0: DIM C%(5), Q%(5), D% (desired
       size): PRINT CHR$(4)"BRUN QUICKI/O"
10  &TI1:T = D%: &AI0:DLAY = 1
20  FOR SAMPLE = 1 TO 10
30  T = T - DLAY
40  &TI1: IF D% > T GOTO 40
50  &AI0,SAMPLE: NEXT SAMPLE
```

ANALYSIS:

Line 10 reads the initial time, T, and discards the first A/D value. The DLAY value is set to 1, which represents one tenth second between samples.

Line 20 loops for 10 samples.

Line 30 subtracts delay time from initial time.

Line 40 waits until delay time is up.

Line 50 reads analog input and stores it in array D% indexed by the loop counter, SAMPLE.

Note that DLAY is a precise time interval measured in tenths of seconds; this is much more accurate than a counting loop written in BASIC. Also note that the loop counter, SAMPLE, is used to index the values stored in D%().

## General Purpose Timers 2 and 3

Timers 2 and 3 are each 16-bit timers that are available for

your use as timers, frequency generators, frequency counters, shift registers, etc. QUICKI/O allows you to set these timers by the &TO2 and &TO3 commands. You can read these timers via the &TI2 and &TI3 commands. Timers 2 and 3 actually represent timers 1 and 2, respectively, on the user 6522 chip (see ADALAB manual). Initially, both are configured as one shot interval timers that count at a rate of 1.023 MHz. However, you can change their mode of operation by POKEing the desired mode value in location $CN3B (or 49211+256*N), where N stands for the slot number of your ADALAB card. See the ADALAB hardware manual for more information about modes of these timers.

## Examples Using Timers 2 and 3

As an example, we'll set up Timer 2 to output a continuous square wave on bit 7 of card # 0's Parallel Output and Timer 3 will count pulses on bit 6 of the Parallel Output. In this example (EX4 on disk), bit 6 of the Parallel Output on card # 0 must be changed to an Input. To do this, the program POKEs the value 191 into location $CN32 (for slot N, type POKE 49202+256*N,191). After the program asks you to enter the TIMER 2 RATE, you may safely jumper bit 7 to bit 6 of the Parallel Output on card # 0 (connect pin 4 to pin 13 on the 16 pin DIP output cable). This allows the pulses output on bit 7 to be input on bit 6.

NOTE: Normally, you should carefully avoid connecting two output signals together on the digital output cable, because if one of the output signals is on while the other is off, an excessive current flow may damage the 6522 chip. On the other hand, connecting two input signals together (to a single external signal) on the digital input cable does no harm. In the present example, line 10 changes bit 6 (normally an output) into an input bit.

To stop this example, you should press <CTRL>C, remove the jumper between bits 6 and 7 and then POKE 49202+256*N,255 or CALL 36096 in order to restore bit 6 to an output bit. Alternatively, you may press <CTRL> RESET to stop the program, but this will require that you CALL 36096 to re-initialize the ADALAB hardware before you start using QUICKI/O functions again.

```
0   REM EXAMPLE 4
1   HIMEM: 36095: D% = 0: DIM C%(5), Q%(5), D% (desired
      size): PRINT CHR$(4)"BRUN QUICKI/O"
5   D% = 255: &PO0:N = PEEK (36222) - 192
10  POKE 49202 + 256 * N,191: POKE 49211 + 256 * N,224
20  INPUT "TIMER 2 RATE?";D%: &TO2
30  &TI3: HTAB 1: PRINT D%;
40  IF PEEK ( - 16384) < 128 GOTO 30
50  GOTO 20
```

ANALYSIS:

Line 5 turns all the bits in the parallel output on. N peeks a location to determine the ADALAB CARD#.

Line 10 sets Data Direction register bit 6 to an input with the first POKE. POKE 49211 + 256 * N, 224 sets timer 2 to the free running mode and eables pulse output on digital output bit 7 each time the counter counts down to zero. Also, timer 3 is set up to count pulses on digital "output" bit 6.

Line 20 allows you to select the pulse rate.

Line 30 reads timer 3 and displays the count.

Line 40 loops until any key is pressed.

## Timers_on_Extra_ADALAB_Cards

If you have more than one ADALAB card, 4 additional timers per board are available for your use. These timers are numbered 4 to 7 for the second card, 8 to 11 for the third card and 12 to 15 for the fourth card. Timers 4, 8 and 12 are initially set up to continuously output pulses at the rate of 10 per second and these pulses are counted by timers 5, 9 and 13, respectively. Thus, timers 5, 9 and 13 may be used exactly the same way as timer 1; that is, as alarm timers. You may change the frequency of timers 4, 8 and 12 with an &TO command. All of these timers count down at the 1.023 MHz clock rate and output one-half pulse to timers 5, 9 or 13 each time the count reaches 0. If you set timers 5, 9 or 13 with an &TO command, the speaker will buzz when their count becomes negative (providing you don't read them with an &TI command before the alarm begins).

Timers 6, 10 and 14 are exactly analogous to timer 2, while timers 7, 11 and 15 are analogous to timer 3. The modes of these timers are entirely up to you, as described above for timers 2 and 3. Consult the ADALAB hardware manual for further information.

## Digital_Input_and_Output

Many people think that Digital I/O is the same as Parallel I/O. However, QUICKI/O makes the distinction that Digital I/O refers to a single bit, whereas Parallel I/O refers to a group of 8 bits. If you have a single ADALAB card, you may use Digital bits 0 to 7. A second card adds bits 8 to 15, a third card adds bits 16 to 23 and a fourth card adds bits 24 to 31. Actually, Digital bits 0 to 7 are the same as the Parallel Output on card # 0 (or input port 0). Likewise, Digital bits 8 to 15 are the same as the Parallel Output on card # 1 (or input port 1), bits 16 to 23 are on card # 2 and bits 24 to 31 are on card # 3. In spite of this overlap in function, our distinction between Digital and Parallel I/O makes it much easier to use ADALAB for turning individual switches on or off (output) and for reading switch states (input). This feature is included in QUICKI/O because Applesoft BASIC is poorly suited for extracting digital (bitwise) information from parallel I/O.

To read a Digital Input bit, type &DI, followed by the bit number. The value returned in D% is 0 if that bit is off. If that bit is on, D% will contain two raised to the power N-(8*C), where N is the bit number and C is the card number of the corresponding Parallel Input port. For example, if bit 9 is on, the value returned in D% will be 2 raised to the 9-(8*1) power (that is, D%=2) because N is 9 and C is 1. Normally, we don't care about the exact value of D%; we only need to determine whether D% is zero or not zero.

To write to a Digital Output bit, set D% to 0 if you want to turn it off or set D% to 1 (or any other nonzero value) to turn it on. Then, issue the &DO command, followed by the desired bit number.

## Examples_Using_Digital_Input/Output

This program will familiarize you with the &DI and &DO commands. Type RUN EX5. Remember, bits can only have a value of 1 or 0.

```
0   REM EXAMPLE 5 - ALSO TEST PROGRAM FOR THE AC/DC POWER RELAY
1   HIMEM: 36095: D% = 0: DIM C%(5), Q%(5), D% (desired
       size): PRINT CHR$(4)"BRUN QUICKI/O"
10  INPUT "INPUT(I) OR OUTPUT(O)?";IO$
20  INPUT "BIT NUMBER?";B
30  IF IO$ = "I" THEN &DIB: PRINT "VALUE="D%
40  IF IO$ = "O" THEN INPUT "VALUE?";D%: &DOB
50  GOTO 10
```

You may stop this program by pressing <CTRL>C.

ANALYSIS:

Line 10 selects input or output.

Line 20 selects the bit #.

Line 30 inputs digital bit selected and prints the value. If the bit is off, the value will be 0; if on, the value will be 2 raised to the bit #.

Line 40 outputs a value to the bit chosen. Any non-zero value will turn the bit on.

## Parallel Input and Output

Parallel I/O means simultaneous Input or Output of 8 bits of information. This is the fastest method for communicating with instruments. A single ADALAB card includes port 0; a second card adds port 1, a third card adds port 2 and a fourth card adds port 3.

To read a Parallel Input value, type &PI, followed by the card number (or port number). The result returned in D% will have a value between 0 and 255. To write a Parallel Output value, put a value in the range of 0 to 255 in D% and then issue the &PO command, followed by the card number. Often, it is convenient to use the D%() array to contain the I/O values. In this case, add a comma and the array element number at the end of the command.

To turn on individual bits 0 through 7, you must place the value for that bit in D% and then issue the &PO0 command:

| BIT | VALUE |
|---|---|
| 0 | 1 |
| 1 | 2 |
| 2 | 4 |
| 3 | 8 |
| 4 | 16 |
| 5 | 32 |
| 6 | 64 |
| 7 | 128 |

For example, to turn on bit 5:

```
D% = 32: & PO0
```

If you wish to turn on multiple bits, add the values for those bits and place that value in D%. For example, to turn on bits 1,2 and 5, add 2 + 4 + 32 = 40; thus, set D% to 40 and issue the command, & PO0.

In order to simplify communication between instruments and the ADALAB board, handshaking signals are provided. These signals help the computer and the instruments to synchronize their information transfer. For example, after sending data, the sender has to make sure that the receiver has received the data. Similarly, a receiver of data has to make sure that the data is valid before accepting it.

When you first run QUICKI/O, it assumes that you don't care about handshaking. In order to make QUICKI/O pay attention to the handshaking signals, you must POKE 36257,1 (for Parallel Input) or POKE 36273,1 (for Parallel Output). To disable handshaking mode, POKE zero in the same locations. If something disturbs the handshaking process (such as disconnecting the cable), the computer may have to wait around forever for data.

QUICKI/O contains a safety measure to prevent this sort of deadlock; if you type any key during Parallel I/O, the handshaking signal will be ignored until the next time your program reads the keyboard.

NOTE: The parallel output handshake signal must be modified in order to work with the ADA-MUX reed relay multiplexer; see the ADA-MUX manual for further details.

Several different types of handshaking can be used with QUICKI/O (see the ADALAB hardware manual for details). To change the handshaking mode, you must POKE the desired MODE value in location $CN3C, where N is the slot number. In BASIC, use POKE 49212+256*N,MODE to do this.

## Examples_Using_Parallel_Input/Output

The following two programs are EX6 and EX7 on the QUICKI/O disk. EX6 allows parallel input or output on any parallel channel:

```
0   REM EXAMPLE 6
1   HIMEM: 36095: D% = 0: DIM C%(5), Q%(5), D% (desired
      size): PRINT CHR$(4)"BRUN QUICKI/O"
10  INPUT "INPUT(I) OR OUTPUT(O)?";IO$
20  INPUT "CARD #?";CH
30  IF IO$ = "I" THEN &PICH: PRINT "VALUE="D%
40  IF IO$ = "O" THEN INPUT "VALUE?";D%: &POCH
50  GOTO 10
```

ANALYSIS:

Line 10 selects Input or Output.

Line 20 selects an ADALAB card. If you have only one ADALAB card, you may change line 20 to read: 20 CH=0.

Line 30 inputs a value.

Line 40 outputs a value.

EX7 inputs 100 values from Parallel Input channel 1 (in slot 2) and stores the values in the D% array:

```
0   REM EXAMPLE 7
1   HIMEM: 36095: D% = 0: DIM C%(5), Q%(5), D% (desired
      size): PRINT CHR$(4)"BRUN QUICKI/O"
10  SLOT = 2: POKE 49212 + 256 * SLOT,8
20  POKE 36257,1
30  FOR I = 1 TO 100
40  &PIO,I: NEXT
```

ANALYSIS:

Line 10 POKEs 8 into the user 6522 Peripheral Control Register for handshaking control.

Line 20 enables handshaking to be recognized by QUICKI/O.

Lines 30 and 40 input 100 samples and store the results in the D% array.

EX8 on disk is a short program that demonstrates parallel input and displays the data in binary, hexadecimal and BCD.

```
0   REM EXAMPLE 8: PARALLEL INPUT OF BINARY, HEXADECIMAL, DECIMAL
      OR BCD DATA
1   HIMEM: 36095: D% = 0: DIM C%(5), Q%(5), D% (desired
      size): PRINT CHR$(4)"BRUN QUICKI/O"
10  &PIO:U = D%: PRINT U,"DECIMAL"
20  U1 = INT (U / 16):U1$ = CHR$ (U1 + 48):V1$ = U1$: IF U1 > 9
      THEN U1 $ = CHR$ (U2 + 55):V1$ = "?"
30  U2 = U - 16 * U1:U2$ = CHR$ (U2 + 48):V2$ = U2$: IF U2 > 9
      THEN U2$ = CHR$ (U2 + 55):V2$ = "?"
40  PRINT V1$;V2$, "BINARY CODED DECIMAL": PRINT
      U1$;U2$,"HEXADECIMAL"
50  FOR I = 7 TO 1 STEP - 1:PWR = 2 ^ I:B = INT (U / PWR): PRINT
      B;
60  U = U - B * PWR: NEXT
70  PRINT U, "BINARY": GOTO 10
80  REM YOU MAY CONVERT THIS PROGRAM INTO A NUMBER INTERCONVERTER
      BY ELIMINATING LINES 0-20 AND ADDING LINE 15
      INPUT"DECIMAL VALUE?";U
```

ANALYSIS:

Line 10 does the Parallel Input and prints the decimal equivalent.

Line 20 calculates the high "nibble" (4-bit) value and converts it to a digit string within the range of 0 to 9. Values from 10 to 15 are converted into letters from A to F. For BCD, these values are illegal, so a question mark is printed instead.

Line 30 similarly computes the low "nibble" value and converts the value to a string.

Line 40 prints the BCD and hexadecimal equivalents.

Line 50 extracts and prints the binary digits from 2^7 to 2^1.

Line 60 subtracts out the current digit value.

Line 70 prints the lowest order bit (2^0) and loops back to input a new value. Press <CTRL-C> to quit.

## Analog Input and Output

An analog signal is a positive or negative voltage. You may select the voltage range for input and output by connecting certain pins on the ADALAB interface card (see the ADALAB hardware manual). Voltage ranges ±0.5V, ±1V, ±2V and ±4V are available. Regardless of which voltage range you select, the "count" value -2047 corresponds to the most negative voltage, the value 0 means zero volts and the value 2047 is the most positive voltage. The card numbers for analog input and output range from 0 to 3, depending on the number of ADALAB cards in your system.

To read an analog voltage, use the &AI command, followed by the card number. Variable D% then contains a value between -2047 and 2047, which denotes the most negative or the most positive voltage, respectively, for the selected range. If D% is -4095 or 4095, this means that the voltage exceeds the limits for accurate voltage conversion. It is important to realize that the analog to digital conversion process takes slightly less than 0.05 seconds; therefore, it is necessary to wait at least this long before asking for a new value. Each time you use the &AI command, the previous reading is returned in D% and a new conversion is started. This approach allows your program to perform some calculations while the next sample is being taken. It also means that the very first sample value taken is inaccurate and should be discarded. Thus, if you want to read a single value, use the &AI command once to start the conversion and then use the &AI command again after 0.05 seconds to read the actual value. If you wish to read more than one value, it is not necessary to issue the &AI command twice for each sample; just bear in mind that the value in D% was sampled immediately following the previous &AI command.

Initially, QUICKI/O is set up to ignore the signal from the A/D converter that tells when a conversion is done. However, if you POKE the value 1 into location 36259 ($8DA3 hex), analog handshaking is enabled and a value will not be returned until the previous conversion is done. This conveniently avoids the need to write a timing loop in BASIC to wait 0.05 seconds between each reading. To return to normal mode (disable analog handshake), POKE 36259, 0.

In order to send a value to the Digital to Analog converter, place a value between -2047 and 2047 in D% and use the &AO command, followed by the appropriate card number. The value -2047 outputs the most negative voltage for the D/A range you have jumper-selected, while the value 2047 outputs the most positive voltage.

NOTE: When the computer is first turned on, the output voltage is undefined, and may be set to the most negative voltage for the selected range; thus, if this negative voltage could harm your instrument, be sure to disconnect the ADALAB cable before turning the computer on.

When QUICKI/O is initialized (by BRUN QUICKI/O), the output

voltage is changed to 0 volts. Thereafter, the output voltage will be determined by your &AO commands and the voltage will remain constant until the next &AO command.

One feature of the Digital to Analog converter that you should know about is that the &AO command automatically triggers the Analog to Digital converter on the same card, as well as the Digital to Analog converter. This will affect you only if you have connected the Analog Output to the Analog Input on the same card. In this case, it is necessary to discard the first two Analog Input values after changing the Analog Output value, due to the delayed response of the Analog to Digital converter. The Digital to Analog converter works much faster than BASIC, and thus, it is not necessary to test for conversion done. (The maximum D/A conversion time is 20 microseconds, including setting time).

## Examples Using Analog Input and Output

Our example EX9 allows input or output of a value on any valid analog channel. Type RUN EX9.

```
0   REM EXAMPLE 9
1   HIMEM: 36095: D% = 0: DIM C%(5), Q%(5), D% (desired
       size): PRINT CHR$(4)"BRUN QUICKI/O"
10  INPUT "INPUT(I) OR OUTPUT(O)?";IO$
20  INPUT "CARD #?";CH
30  IF IO$ = "I" THEN &AICH: FOR I = 1 TO 50: NEXT : &AICH:
       PRINT "VALUE= "D%
40  IF IO$ = "O" THEN INPUT "VALUE ";D%: &AOCH
50  GOTO 10
```

ANALYSIS:

Line 10 selects Input or Output.

Line 20 selects an ADALAB card. If you have only one card, change line 20 to read: 20 CH=0.

Line 30, if Input, reads and discards one analog input, waits 50 milliseconds before reading another analog input, and then prints this value.

Line 40, if Output, the user supplies a value and this is output on the ADALAB card selected.

## Analog Input at Maximum Sampling Rate

The next program takes 280 Analog Input samples on card # 0 at the maximum rate permitted by the conversion done signal. Each reading is scaled for plotting on the high resolution screen, with the value 2047 at the top of the screen and -2047 at the bottom of the screen. At the end, the graph remains on the

screen until you press any key. This is program EX10 on disk.

```
0   REM EXAMPLE 10
1   HIMEM: 36095: D% = 0: DIM C%(5), Q%(5), D% (desired
      size): PRINT CHR$(4)"BRUN QUICKI/O"
10  &AI0: POKE 36259,1:HGR
20  FOR SAMPLE = 0 TO 279
30  &AI0:Y = 96 - D% / 21.5
40  HPLOT SAMPLE,Y: NEXT SAMPLE
50  IF PEEK ( - 16384) , 128 GOTO 50
60  TEXT
```

ANALYSIS:

Line 10 discards the initial reading, enables analog handshake and erases the graphics screen.

Lines 20 and 30 input 280 analog signals and scale the data.

Line 40 plots the scaled signal and continues.

Line 50 waits until a key is pressed to erase the graph.

Line 60 returns to TEXT mode.

## Timing Analog Input Samples

One disadvantage of Example 9 is that the time required for the A/D converter to read a voltage varies with the voltage. Voltages close to 0 are measured faster than voltages near the full scale positive or negative value. For this reason, we recommend that you use Timer 1 to wait for a fixed time interval between samples, as in Example 3. To implement a fixed sampling rate of 10 samples per second, you may add these lines to Example 9:

```
15 &TI1: T=D%: DLAY=1
25 T= T-DLAY
28 &TI1: IF D%>T GOTO 28
```

Alternatively, you may introduce a fixed delay between samples by including a FOR..NEXT loop, as follows:

```
15 DLAY=50
25 FOR I=1 TO DLAY: NEXT
```

Set DLAY to any value greater than 50; the delay will be about 1 millisecond per count. For example, DLAY=50 would yield a delay of about 50 milliseconds or a sampling rate of about 20 samples per second.

## Analog_Output_of_Sawtooth_Waveforms

The following program outputs a sawtooth wave form on the Analog Output on card # 0. You select the frequency from 0.01 to 100 waves per second (HZ). This frequency is only approximate, but may be adjusted by changing the value of SCAL in line 10. For each frequency, the output voltage ranges from 0 volts to the maximum for the selected voltage range. With the faster speeds, an oscilloscope trace will show a more jagged pattern, due to the large step size. This is file EX11 on the disk.

```
0   REM EXAMPLE 11
1   HIMEM: 36095: D% = 0: DIM C%(5), Q%(5), D% (desired
       size): PRINT CHR$(4)"BRUN QUICKI/O"
10  MN = 0:MAX = 2047:SCAL = 20
20  INPUT "FREQUENCY (0.01 TO 100 HZ)?";FRQ
30  DV = FRQ * SCAL
40  FOR V = MN TO MAX STEP DV
50  D% = V: & AO0: NEXT V
60  IF PEEK (- 16384) < 128 GOTO 40
70  GOTO 20
```

ANALYSIS:

Line 10 sets the voltage range from 0 to the maximum positive voltage selected by the jumpers on the ADALAB card.

Line 20 selects the frequency of the saw tooth wave.

Line 30 calculates the step size.

Line 40 loops from minimum to maximum voltage.

Line 50 has the loop counter as the output voltage, performs analog output of the voltage and continues.

Line 60 continues until any key is pressed.

Line 70 returns to the frequency selection quesiton. You should press <CTRL>C to stop this program.

The final example reads the A/D converter and prints the corresponding voltage, depending on the jumper range selected on the ADALAB card. TYPE RUN EX12.

```
0   REM EXAMPLE 12
1   HIMEM: 36095: D% = 0: DIM C%(5), Q%(5), D% (desired
       size): PRINT CHR$(4)"BRUN QUICKI/O"
10  REM ENABLE ANALOG HANDSHAKE
11  POKE 36259,1
20  INPUT "FULL SCALE A/D RANGE (0.5,1,2,4)?";RANGE
30  REM DISCARD FIRST READING
35  &AI0
40  &AI0: PRINT "VOLTAGE = "D% / 2047 * RANGE;
50  HTAB 1: GOTO 40
```

ANALYSIS:

Line 11 defines Analog handshaking.

Line 20 establishes the voltage corresponding to a full-scale reading (±2047).

Line 35 discards the first A/D value.

Line 40 reads a second A/D value and prints the value after converting to a voltage.

Line 50 loops back and repeats the A/D conversion continuously; press <CTRL-C> to stop the program.

## USING THE ADALAB REAL-TIME CLOCK WITH BASIC PROGRAMS

Normally, the &T commands in QUICKI/O are ideally suited for setting and reading ADALAB's real-time clock and timers. However, if QUICKI/O conflicts with the memory space occupied by another program or if you want to minimize the amount of memory space devoted to ADALAB routines, you may use one of the alternate programs described here.

The first and simplest program (ADALAB TIMER I/O #1, on page E-3) uses Timer 1 to count tenths of seconds and does not use interrupts. To set the time, set D% to a number (in tenths of seconds) and CALL 880 ($370). To read the time, CALL 908 ($38C), and the current time is returned in D% (in tenths of seconds). To wait for a given time, set D% to the desired time (in tenths of seconds) and CALL 921 ($399). Your program will resume when Timer 1 counts down to that value. In case of error (such as setting the desired time greater than the current time), your program may halt for as long as 1.82 hours. In this unfortunate case, you may terminate the request by pressing any key on the keyboard. The accompanying BASIC program is an example of how you can use this program to SET, GET, or WAIT for a given time and convert readings from timer 1 into hours, minutes, and seconds.

Current versions of the QUICKI/O disk contain this program as file TIMOBJ1, and the BASIC test program is called TIMTEST1.

You will note that TIMOBJ1 resides just above the ADOBJ machine language routine (described in the ADALAB hardware manual) for reading ADALAB's A/D converter.

The machine language subroutine listed on the next page allows you to use the ADALAB A/D converter from a BASIC program. To use this program with your BASIC program you must type BSAVE ADOBJ,A$320,L$50. Your BASIC program must begin with LOMEM:24576: D%=0 to ensure that the value of D% is stored at the right place. To read the A/D value, set D% to the slot number of your ADALAB card and CALL 800. On return, D% contains the current A/D value, sampled after the appropriate time delay. The most negative voltage reading is -4095 and the most positive voltage is 4095. Over-range is indicated by -8192 or 8192. Thus, you can use both the A/D routine and the timer routine in the same program. On the QUICKI/O disk, the ADOBJ and TIMOBJ1 programs are combined into one program, ADTIM. This combined program uses Timer 1 to accurately control the delay between samples. Since the timer runs independently of your BASIC program, you can perform some calulations between samples in the BASIC program and still maintain an accurate time interval between samples.

The second time-keeping program (ADALAB TIMER I/O #2, on page E-4) uses 50 millisecond interrupts to update the time in hours, minutes, and seconds. To initialize the real-time clock, CALL 880 ($370). To set the time, POKE HOURS at 972 ($3CC), POKE MINUTES at 971 ($3CB), POKE SECONDS at 970 ($3CA), and POKE UNITS at 969 ($3C9) where UNITS are measured as 50 millisecond intervals. To read the time, read HOURS, MINUTES, SECONDS, and UNITS at the same corresponding locations. Here, the time is automatically updated, so you don't need to calculate the time from the 50 millisecond down-counter value. The accompanying BASIC program gives an example of how to SET or GET the time and WAIT for a particular time to arrive. It also shows how to display a continuously updated time on the video screen.

Current versions of the QUICKI/O disk contain this program as file TIMOBJ2 and the BASIC test program is called TIMTEST2.

Both TIMOBJ1 and TIMOBJ2 assume that your ADALAB interface card is in slot 2. If this is not the case, you should change all of the $C2's in the 3-byte instructions to $CN, where N is the slot containing the ADALAB card. Also, note that both TIMOBJ1 and TIMOBJ2 are relocatable to any address in memory, although if you move TIMOBJ2, you must update INTADR (location $3C7) to contain the new address of the TIMINT interrupt routine.

```
                        0010  ;ADALAB TIMER I/O#1
                        0020  DPER    EQU 6002
                        0030  BASE1   EQU C200
                        0080          ORG 0370
                        0085          OBJ 8000
                        0090  ;SET TIME IN D%
0370   A9E0     0100  SETIM   LDA #$E0
0372   8D08C2   0110          STA BASE1+0B
0375   A9BE     0120          LDA #$BE
0377   8D04C2   0130          STA BASE1+04
037A   A9C7     0140          LDA #$C7
037C   8D05C2   0150          STA BASE1+05
037F   AD0360   0160          LDA DPER+01
0382   8D08C2   0170          STA BASE1+08
0385   AD0260   0180          LDA DPER
0388   8D09C2   0190          STA BASE1+09
038B   60       0200          RTS
                        0210  ;GET TIME IN D%
038C   AD08C2   0220  GETIM   LDA BASE1+08
038F   AC09C2   0230          LDY BASE1+09
0392   8D0360   0240          STA DPER+01
0395   8C0260   0250          STY DPER
0398   60       0260          RTS
                        0270  ;WAIT TIME IN D%
0399   2C00C0   0280  WATIM   BIT $C000
                        0285  ;TIMEUP ON KEYPRESS
039C   3010     0290          BMI OUT
039E   AD08C2   0300          LDA BASE1+08
03A1   CD0360   0310          CMP DPER+01
03A4   D0F3     0320          BNE WATIM
03A6   AD09C2   0330          LDA BASE1+09
03A9   CD0260   0340          CMP DPER
03AC   D0EB     0350          BNE WATIM
03AE   60       0360  OUT     RTS
```

```
LABEL TABLE
DPER    6002
BASE1   C200
SETIM   0370
GETIM   038C
WATIM   0399
OUT     03AE
```

```
]LIST

1  LOMEM: 24576:D% = 0: PRINT  CHR$
   (4)"BLOAD TIMOBJ1,A$370"
100  INPUT "SET(S), GET(G) OR WAI
   T(W) ?";A$
110  IF A$ = "S" THEN  GOSUB 9000

120  IF A$ = "G" THEN  GOSUB 9100

130  IF A$ = "W" THEN  GOSUB 9200

140  GOTO 100
9000 SETIM = 880:GTIM = 908:WTIM =
   921:D% = 32767: CALL SETIM
9010  INPUT "TIME (HOURS,MIN,SEC)
   ?";HR,MN,SC: CALL GTIM:T0 =
   D%: RETURN
9100  CALL GTIM:T1 = D%:DT = T0 -
   T1: IF DT < 0 THEN DT = DT +
   65536
9110 D% = DT / 36000:DT = DT - D%
    * 36000:HR = HR + D%
9120 D% = DT / 600:DT = DT - D% *
   600:MN = MN + D%
9130 D% = DT / 10:DT = DT - D% *
   10:T0 = T1 + DT:SC = SC + D%

9140  IF SC >  = 60 THEN SC = SC -
   60:MN = MN + 1
9150  IF MN >  = 60 THEN MN = MN -
   60:HR = HR + 1
9160  PRINT "TIME IS "HR":"MN":"S
   C","DT: RETURN
9200  INPUT "DELAY TIME (TENTHS O
   F SECONDS) ?";DLY
9210  CALL GTIM:D% = D% - DLY: CALL
   WTIM: RETURN
```

```
]LIST 1010-

1010  IF AD = 0 THEN AD = 800: PRINT
   CD$"BLOAD AD$(IM,A"AD
1012 T = 32767:D% = T: CALL 880:D
   T = 921
1015  FOR I = 0 TO NS:D% = DP: CALL
   AD:DX(I,80) = D%:U =  USR (N
   ):T = T - DLY:D% = T: CALL W
   T: NEXT :U =  USR (A): TEXT
   : RETURN
```

```
                0010  ;ADALAB TIMER I/O#2
                0020  DPER    EQU 6002
                0030  BASE1   EQU C200
                0080          ORG 0370
                0085          OBJ 8000
                0090  ;INITIALIZE TIMER 0
0370  A9E0      0100  SETIM   LDA #$E0
0372  8D0BC2    0110          STA BASE1+0B
0375  A9BE      0120          LDA #$BE
0377  8D04C2    0130          STA BASE1+04
037A  A9C7      0140          LDA #$C7
037C  8D05C2    0150          STA BASE1+05
037F  78        0160          SEI
0380  ADC703    0170          LDA INTADR
0383  8DFE03    0180          STA $03FE
0386  ADC803    0190          LDA INTADR+0
                    1
0389  8DFF03    0200          STA $03FF
038C  A9C0      0210          LDA #$C0
038E  8D0EC2    0220          STA BASE1+0E
0391  58        0230          CLI
0392  60        0240          RTS
0393  98        0250  TIMINT  TYA
0394  48        0260          PHA
0395  AD04C2    0270          LDA BASE1+04
0398  EEC903    0280          INC UNITS
039B  ADC903    0290          LDA UNITS
039E  C914      0300          CMP #$14
03A0  3020      0310          BMI TIMOK
03A2  A900      0320          LDA #$00
03A4  8DC903    0330          STA UNITS
03A7  EECA03    0340          INC SECS
03AA  A03C      0350          LDY #$3C
03AC  CCCA03    0360          CPY SECS
03AF  D011      0370          BNE TIMOK
03B1  8DCA03    0380          STA SECS
03B4  EECB03    0390          INC MINS
03B7  CCCB03    0400          CPY MINS
03BA  D006      0410          BNE TIMOK
03BC  8DCB03    0420          STA MINS
03BF  EECC03    0430          INC HOURS
03C2  68        0440  TIMOK   PLA
03C3  A8        0450          TAY
03C4  A545      0460          LDA $45
03C6  40        0470          RTI
03C7  9303      0500  INTADR  EQD TIMINT
03C9  00        0510  UNITS   DFD 00
03CA  00        0520  SECS    DFD 00
03CB  00        0530  MINS    DFD 00
03CC  00        0540  HOURS   DFD 00
```

```
LABEL TABLE
DPER   '6002
BASE1   C200
SETIM   0370
TIMINT  0393
TIMOK   03C2
```

```
]LIST

1  LOMEM: 24576:D% = 0: PRINT  CHR$
     (4)"BLOAD TIMOBJ2,A$370"
100  PRINT "SET(S), GET(G) OR WAI
     T(W) ?";
105  IF  PEEK ( - 16384) < 128 THEN
      GOSUB 9300: GOTO 105
108  GET A$: PRINT A$
110  IF A$ = "S" THEN  GOSUB 9000

120  IF A$ = "G" THEN  GOSUB 9100

130  IF A$ = "W" THEN  GOSUB 9200

140  GOTO 100
9000 SETIM = 880:ULOC = 969:SLOC =
     970:MLOC = 971:HLOC = 972: CALL
     SETIM
9010  INPUT "TIME (HOURS,MIN,SEC)
      ?";HR,MN,SC: POKE HLOC,HR: POKE
     MLOC,MN: POKE SLOC,SC: POKE
     ULOC,0: RETURN
9100  PRINT "TIME IS " PEEK (HLOC
     )":" PEEK (MLOC)":" PEEK (SL
     OC)"." PEEK (ULOC): RETURN
9200  INPUT "DELAY UNTIL (HOURS,M
     IN,SEC) ?";HR,MN,SC
9210  IF  PEEK (HLOC) < HR OR  PEEK
     (MLOC) < MN OR  PEEK (SLOC) <
     SC THEN  GOSUB 9300: GOTO 92
     10
9220  PRINT : RETURN
9300 HT =  PEEK (36): PRINT  PEEK
     (HLOC)":" PEEK (MLOC)":" PEEK
     (SLOC)"." PEEK (ULOC)" ";: POKE
     36,HT: RETURN
```

## USING THE ADALAB A/D CONVERTER WITH CURVE FITTER

To modify CURVE FITTER so that the Control Y command reads the ADALAB A/D converter, type LOAD CURFIT and make the following changes exactly. Then, type LOAD CURFITAD to save this new version.

```
10 HIMEM:38399: LOMEM:24576: D%=0: DIM IN$(50), D(1000), DD(5,2):
MX=1000

2900 IF AD=0 THEN SLOT=2: AD=800: PRINT: PRINT CD$"BLOAD
ADOBJ,A"AD

2910 D%=SLOT: CALL AD: V0=D%: RETURN
```

# MODIFICATIONS OF QUICKI/O TO AVOID INTERFERENCE WITH THE MICROSOFT Z-80 SOFTCARD

During initialization of QUICKI/O, the program addresses each interface slot of the Apple computer to determine whether an ADALAB card resides there. Unfortunately, when a Z-80 Softcard is used, this turns on the Z-80 processor and leaves you in limbo. To avoid this problem, BLOAD QUICKI/O,A$8D00 and then CALL-151 to enter the monitor. At location $8D7D, you will enter the number of ADALAB cards you have in your system and at $8D7E and thereafter, you will enter the numbers of the slots they occupy, plus $C0. For example, if you have two ADALAB cards in slots 1 and 4, type the following:

```
8D7D: 02 C1 C4
```

Also, you should enter the following patch, which eliminates the search routine:

```
93AD: A9 4C 85 EB 4C E1 93
```

Now, save this modified version of QUICKI/O by typing:

```
BSAVE QUICKI/O,A$8D00,L$8F8
```

Use this version whenever you are using ADALAB together with a Z-80 softcard. If you move ADALAB to a different slot, remember to change QUICKI/O as described above.